IEEE Std 1609.2-2016 Guidance Note 2: Certificate Chains

William Whyte, Security Innovation

Version 0.4, 2016-08-12

1 Introduction

This note identifies an area in IEEE Std 1609.2-2016 that has been reported as potentially misleading or ambiguous and gives guidance on a recommended interpretation of the standard.

The potentially misleading or ambiguous text concerns the length of certificate chains that CAs can issue.

This guidance note is structured as follows. Section 2 provides a summary of the issue, the impact, and current practice. Section 3 provides the recommended interpretation. Section 4 provides potential replacement text that should be considered for incorporation into a revision or amendment of 1609.2.

2 Summary of issue

2.1 Background: Certificate chains, minChainDepth field, rationale for current design

IEEE 1609.2 supports certificate chains, where a message is signed by a certificate, which is in turn issued by another certificate, which is in turn issued by another certificate, and so on until a certificate is reached which was issued by itself and is known as the root certificate. The certificate that signed a message is known as an end-entity certificate. A certificate that issues another certificate is known as a CA certificate. A CA certificate that issues an end-entity certificate is known as a Pseudonym CA certificate (there are other types of CA that issue end entity certificates, but for purposes of this note we focus on pseudonym certificates). This is illustrated in Figure 1.

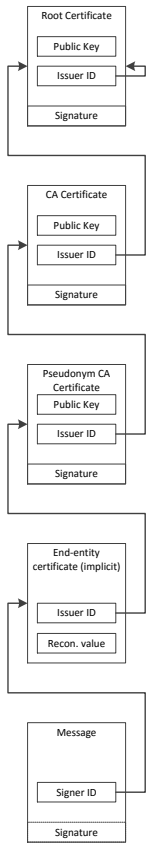


Figure 1: Certificate Chain

CA certificates are subject to constraints on the length of the chain "below" them, i.e. how many other CA certificates can be between a given CA certificate and an end-entity certificate. These constraints are expressed as the two values *minChainDepth* and *chainDepthRange* in the appropriate fields in the CA certificate. The length of the chain from

the CA certificate to the end-entity certificate must be at least *minChainDepth*, and it must be no more than *minChainDepth* + *chainDepthRange*. This allows higher-level CAs to state in their certificates that they only issue CA certificates, which protects them against needing to be online and have the scalability needed to manage end-entity certificates. It also allows pseudonym CAs to state that they only issue certificates for end-entities: this means that if they are compromised the attacker can only issue end-entity certificates, not additional CA certificates, which reduces the impact of compromising one of these high-availability, always-online components.

2.2 Issue

The issue is that the "depth" referred to by minChainDepth is ambiguously defined. To give a concrete example, it is not clear what values should appear for minChainDepth and chainDepthRange in a Pseudonym CA certificate.

- 1609.2 clause 5.1.2.1 says "The *length of the chain* is defined as the number of certificates from the certificate under consideration to the root, inclusive of both (so if the root were to sign a SPDU, the associated certificate chain length would be 1)." This implies that **the length of a chain from a Pseudonym CA to an end-entity is 2** (NOTE however that this clause of 1609.2 is careful to define the length of a chain only with respect to the root; note also though that this clause doesn't draw attention to this fact or provide any guidance about how the length of a chain should be defined if it doesn't include the root. NOTE also that this section uses the term "length" while the other sections discussed below use the term "depth" – this distinction however may not be clear to the reader).
- 1609.2 clause 5.1.2.4 says that an end-entity certificate carries implicit minChainDepth and chainDepthRange values of 0, and: "if *minChainDepth* and *chainDepthRange* in the subordinate certificate and issuing certificates have the values mcd_s, cdr_s, mcd_i, cdr_i, respectively, then $mcd_i \leq mcd_s+1$ and $(mcd_i + cdr_i) \geq (mcd_s + cdr_s +1)$". **This implies that the "depth" of the chain from a pseudonym CA certificate to an end-entity certificate is 1**, and this can be permitted by setting *minChainDepth* and *chainDepthRange* in the Pseudonym CA certificate to 0 and at least 1, respectively, or by setting them 1 and at least 0, respectively.
- 1609.2 clause 6.4.30 says "The length of the certificate chain is measured from the certificate issued by this certificate to the end-entity certificate … The length is permitted to be (a) greater than or equal to `minChainDepth` certificates and (b) less than or equal to `minChainDepth + chainDepthRange` certificates." This is ambiguous and, in the absence of an unambiguous definition of length of a chain to a non-root, could imply that **the "depth" of the chain from a pseudonym CA certificate to an end-entity certificate is 0 or 1**. Figure 2 illustrates possible interpretations of this clause for the case where *minChainDepth* = 3.
- 1609.2 clause 6.4.30 also says, in the part covered by the ellipsis in the previous bullet point, "a length of 0 therefore indicates that the certificate issued or requested is an end-entity certificate." This states that **the "depth" of the chain from a pseudonym CA certificate to an end-entity certificate is 0**.

- 1609.2 clause 6.4.30 also says, in the examples, "A certificate for a CA that issues authorization certificates, i.e., certificates containing an appPermissions field (see 5.1.1), might have an instance of this field for a given PSID/SSP combination with `minChainDepth` equal to 1, `chainDepthRange` equal to 0". This is the same as saying "A pseudonym CA certificate … might have an instance of this field for a given PSID/SSP combination with `minChainDepth` equal to 1, `chainDepthRange` equal to 0". **This states that the "depth" of the chain from a pseudonym CA certificate to an end-entity certificate is 1**.
- 1609.2 clause 6.4.30 also states in the ASN.1 that the default values of minChainDepth and ChainDepthRange are 1 and 0, respectively. The default values are the ones that are intended to be used most often, and since pseudonym CAs are at the bottom of the hierarchy there will more pseudonym CAs than any type of CAs, so this is evidence that the intent was for 1 and 0 to be the values used by pseudonym CAs, i.e. that **the depth for the chain from a pseudonym CA certificate to an end-entity certificate is 1**.

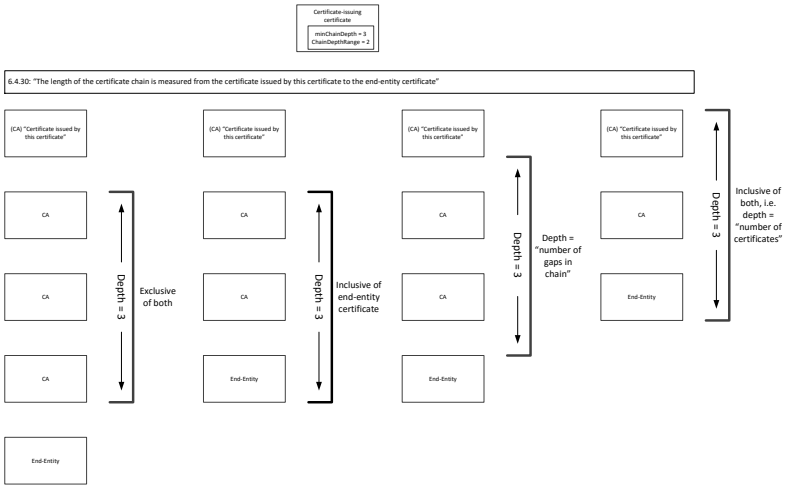


Figure 2 -- different possible meanings of "depth" in 1609.2 clause 6.4.30

2.3 Impact

As noted above, an implementer could potentially interpret the standard such that in a pseudonym CA certificate that can only issue end-entity certificates, the correct value for minChainDepth is 0 or 1 or 2.

This creates a potential for valid messages being rejected. If the creator of a pseudonym CA certificate has set minChainDepth to 0 (and chainDepthRange to 0), but the receiver of a message expects the pseudonym CA certificate to have minChainDepth set to 1 (and

chainDepthRange set to 0), then the receiver will reject the certificate chain because it is not consistent by the definition of clause 5.1.2.4 and so will reject the message.

To avoid valid messages being rejected it is necessary for unambiguous guidance to be given as to the correct interpretation of minChainDepth.

2.4 Existing implementations

There are a number of existing implementations known to this writer. They are all consistent with each other and all use minChainDepth = 1, chainDepthRange = 0 for PCA certificates.

- The CAMP SCMS certificate profile for pseudonym CA certificates (https://stash.campllc.org/projects/SCMS/repos/scms-asn/browse/cert-profile.asn) uses minChainDepth = 1, chainDepthRange = 0.
- The Security Innovation Aerolink implementation follows the CAMP SCMS certificate profile.
- The Green Hills implementation of the CAMP SCMS follows the CAMP SCMS certificate profile

This writer does not know of any implementations that make different interpretations.

3 Recommended interpretation

The recommended interpretation of 1609.2 is: When calculating the depth of a chain for purposes of establishing that a certificate was validly issued per the minChainDepth and chainDepthRange parameters in a particular CA certificate *C*, the depth is the number of certificates in the chain, **including** the end-entity certificate but **excluding** the CA certificate *C*. For example, the depth of the chain from a pseudonym CA to the end-entity certificate that it issues is 1.

4 Recommended change to the standard

This section presents proposed changes to the standard. The changes are presented as marked-up changes to the current text. Any text below in ***bold italics*** is editorial instructions and not part of the proposed text. Cross-references to other subclauses of 1609.2 are highlighted in grey as a reminder to the 1609.2 editor to replace them with Word cross-references when these changes are made to the standard.

These changes also tidy up the use of “length” versus “depth” and introduce the term “full chain” for the chain back to a root.

5.1.1 Certificate contents

Change the third bullet point in the third bulleted list from

— The permissible length(s) of the certificate chain from the certificate containing these issuance permissions to the certificate that signs the PDU. This length is referred to as the permitted depth. Certificate chain length is defined in 5.1.2.1.

to

— The permissible length(s) of the certificate chain from the certificate containing these issuance permissions to the certificate that signs the PDU. The words "length" and "depth" are used synonymously in this standard. Certificate chain length is defined in 5.1.2.1.

5.1.2.1 Certificate chain construction

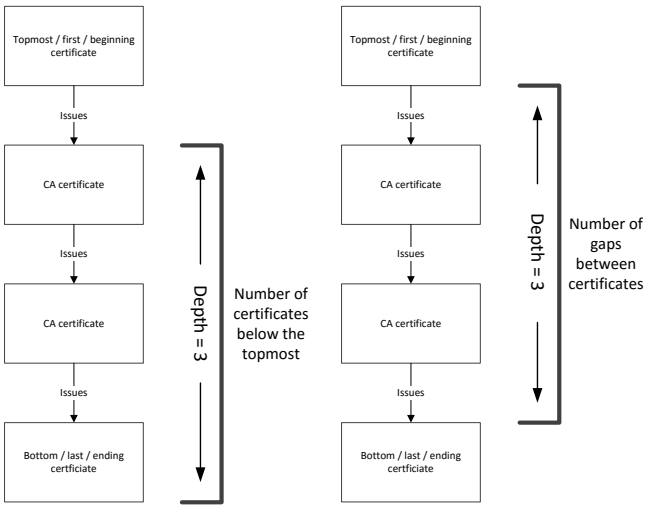
A *certificate chain* is a set of certificates ordered from "top" to "bottom", (equivalently, "first" to "last" or "beginning" to "end") such that each certificate in the chain is the *issuing certificate* for one below (or "after") it and the *subordinate certificate* of the certificate above (or "before") it.

One certificate is the *issuing certificate* for a second one if the certificate holder of the first certificate used the private key of the first certificate to create the final form of the second certificate, either by signing it (in the case of an explicit certificate) or by carrying out cryptographic operations to create a reconstruction value (in the case of an implicit certificate). The counterpart of an issuing certificate is a *subordinate certificate*. If certificate B is the issuing certificate for certificate A, for compactness this ~~certificate~~ standard uses the terminology "B issues A" even though it would be more correct to use the terminology "B's holder issues A" or "the private key associated with B issues A".~~.~~

A *trust anchor* is any certificate that is established to be trustworthy by itself, e.g., by preconfiguration or independent provisioning; in other words, not by reference to any other certificate. A necessary (but not sufficient) condition for a certificate to be valid is that it is possible to construct a certificate chain from the certificate to a trust anchor. The SSME stores information about which certificates are trust anchors.

A *root certificate* is an explicit certificate that is verified with the public key included directly in the certificate, in contrast to other certificates that are verified using the verification key of the issuing certificate. There is no distinct issuing certificate for a root certificate. All trusted root certificates are by definition trust anchors. A certificate chain that starts with a root certificate is referred to in this standard as a *full certificate chain*.

The *length of* ~~*the*~~ *a certificate chain* is defined as the number of certificates in the chain apart from the topmost one, or equivalently as the number of intra-certificate gaps in the chain~~from the certificate under consideration to the root, inclusive of both (so if the root were to sign a SPDU, the associated certificate chain length would be 1). This is true even if the local trust anchor used by a particular implementation is not the root but some intermediate CA~~. These two definitions are illustrated in Figure XXX. In this standard, "length" and "depth" of a certificate chain are used synonymously.



Field Code Changed

Figure XXX – A certificate chain of length 3.

To enable construction of certificate chains, each IEEE 1609.2 certificate contains an identifier for its issuer. An example algorithm for the construction of certificate chains is specified in D.3.1 and illustrated in D.3.2. In constructing the certificate chain, the receiver uses a combination of the certificates that were included in the signed SPDU and locally cached copies of certificates. Local copies of certificates are managed by the SSME.

The issuer identifier is obtained by calculating an eight-octet cryptographic digest of the issuing certificate. There is therefore a 2^{-64} probability that any pair of CA certificates known to the SSME have the same eight-byte digest. If this happens, then when a certificate or SPDU comes to be validated, two chains can be constructed consistent with the issuer identifiers. The certificate or SPDU is considered valid if either of the chains is cryptographically valid.

5.1.2.2 Maximum supported certificate chain length

An implementation of WAVE Security Services may have a maximum length of full certificate chain that it can support. A conformant implementation shall support a maximum length of at least ~~eight~~seven, i.e. a maximum total number of certificates in the chain of at least eight. The Protocol Implementation Conformance Statement (PICS) proforma given in Annex A allows the vendor of an implementation of WAVE Security Services to state the maximum length of certificate chain that the implementation supports.

5.1.2.2 Maximum supported certificate chain length

Change the caption to figure 5 from

Cryptographic verification with explicit certificates

To

Cryptographic verification of a signed SPDU and a full certificate chain, using explicit certificates

Change the caption to figure 6 from

Length-2 certificate chain with implicit certificates and a non-root CA as trust anchor

To

Cryptographic verification of a signed SPDU and a (non-full) certificate chain with implicit end-entity certificate

5.1.2.4 Consistency of permissions within a certificate chain

The relevant material is the third sub-bullet under "consistency of application/issuance permissions", which does not need to be changed. It is reprinted below for ease of reference.

— If the subordinate entry is for cert issuance or request permissions, the permitted depth of the chain in the subordinate entry is consistent with the permitted depth of the chain in the issuing entry. Specifically, if *minChainDepth* and *chainDepthRange* in the subordinate certificate and issuing certificates have the values mcd_s, cdr_s, mcd_i, cdr_i, respectively, then $mcd_i \leq mcd_s+1$ and $(mcd_i + cdr_i) \geq (mcd_s + cdr_s + 1)$. (In the case where the subordinate certificate is an end-entity certificate, mcd_s and cdr_s are set equal to zero (0) in these formulas.)

Add the following clause.

5.1.2.5 Trustworthiness of a certificate chain

A certificate chain is trustworthy if both of the following hold:

— Each subordinate certificate is consistent with its issuing certificate.

— The chain begins with a trust anchor.

5.1.3.2 Determining which revocation information applies to a given certificate

Change

The CRACA certificate for a certificate *C* is only valid if it is on *C*'s issuing chain.

To

The CRACA certificate for a certificate *C* is only valid if it is one of the certificates in *C*'s full chain.

5.1.3.3 Identification of CRACA certificate

Replace both occurrences of "chain" ***with*** "full chain".

5.1.3.6 Dubious certificates

Replace all four occurrences of "chain" ***with*** "full chain".

5.2.3.3.1 General (first subclause of 5.2.3.3 SDEE-specific consistency conditions)

In the list of SDEE-specific consistency conditions, change the fourth bullet as follows:

— (Optionally) The number of certificates in the full chain ending in the SPDU-signing~~, from end-entity certificate to root certificate inclusive~~ certificate~~.~~ is less than some SDEE-specific limit (see Annex C). This condition can be verified by the SDS.

5.2.4.2.1 General (first subclause of 5.2.4.2 SDS-verified relevance conditions)

In the list of SDS-verified relevance conditions, change the sixth bullet as follows:

— **Certificate expiry**: None of the certificates in the full chain ~~leading~~ ending with~~to~~ the certificate that signed the signed SPDU ~~signed data~~ have expired.

5.2.4.2.7 Certificate expiry

Replace both occurrences of "chain" ***with*** "full chain".

6.4.30 PsidGroupPermissions

Remove the comment following minChainDepth in the ASN.1 definition:

```
PsidGroupPermissions ::= SEQUENCE {
    subjectPermissions SubjectPermissions,
    minChainDepth      INTEGER DEFAULT 1, -- 0 for enrolment certs
    chainDepthRange    INTEGER DEFAULT 0, -- max depth = min + range
    eeType             EndEntityType DEFAULT {app}
}

SequenceOfPsidGroupPermissions ::= SEQUENCE OF PsidGroupPermissions
```

Change the description of minChainDepth and chainDepthRange as follows:

— `minChainDepth` and `chainDepthRange` indicate how long the certificate chain from this certificate to the end-entity certificate is permitted to be. As specified in 5.1.2.1, t~~T~~he length of the certificate chain is the number of certificates "below" this certificate in the chain, down to and including the end-entity certificate, ~~measured from the certificate issued by this certificate to the end-entity certificate in the case of `certIssuePermissions` and from the certificate requested by this certificate to the end-entity certificate in the case of `certRequestPermissions`; a length of 0 therefore indicates that the certificate issued or requested is an end-entity certificate.~~ The length is permitted to be (a) greater than or equal to `minChainDepth` certificates and (b) less than or equal to `minChainDepth` + `chainDepthRange` certificates. A value of 0 for `minChainDepth` is not permitted when this type appears in the `certIssuePermissions` field of a `ToBeSignedCertificate`; a certificate that has a value of 0 for this field is invalid. The value −1 for `chainDepthRange` is a special case: if the value of `chainDepthRange` is −1 that indicates that the certificate chain may be any length equal to or greater than `minChainDepth`. See the examples below for further discussion.

Change the Examples as follows:

Examples:

— An enrolment certificate contains a `certRequestPermissions` field containing ~~has~~ an instance of this ~~field~~ type with `minChainDepth` equal to 0, `chainDepthRange` equal to 0, and `eeType` equal to `app` (because the enrolment certificate is used to request authorization certificates).

— A certificate for a CA that directly issues ~~authorization~~ end-entity certificates~~, i.e., certificates containing an appPermissions field (see 5.1.1),~~ might ~~have~~ contain a `certRequestPermissions` field containing an instance of this ~~field~~ type for a given PSID/SSP combination with `minChainDepth` equal to 1, `chainDepthRange` equal to 0, and `eeType` equal to `app`. This indicates that it is entitled to issue end-entity certificates for ~~those~~ that PSID/SSP combinations ~~but not to sign application messages~~.

— A certificate for an intermediate CA might contain a `certRequestPermissions` field containing ~~have~~ an instance of this ~~field~~ type for a given PSID/SSP combination with `minChainDepth` equal to 2, `chainDepthRange` equal to 0, and `eeType` equal to `app`. This indicates that there must be exactly one CA in the chain between the intermediate CA and the end-entity. ~~it is entitled to issue EECA certificates for those PSIDs but not to issue end-entity certificates directly.~~

— A certificate for a root CA might have an instance of this field for a given PSID/SSP combination with `minChainDepth` equal to 3, `chainDepthRange` equal to −1, and `eeType` equal to (`app`, `enrol`). This indicates that there must be at least two CAs in the chain between the root CA and the end-entity (`minChainDepth` = 3) and that there may be any number greater than or equal to two (`chainDepthRange` = −1, i.e. the length of the chain is not constrained so long as it is greater than or equal to `minChainDepth`). ~~it is entitled to issue ICA certificates and that these ICA certficates are entitled to appear in chains that lead to both authorization certificates and enrolment certificates.~~

— The certificate for the root CA used to support the Connected Vehicle Pilot Deployments has the following certIssuePermissions field (presented using ASN.1 value notation):

```
    certIssuePermissions (SequenceOfPsidGroupPermissions (SIZE(4)) (CONSTRAINED BY {
      PsidGroupPermissions ( WITH COMPONENTS {...,
            subjectPermissions (WITH COMPONENTS {
          all
        }),
        minChainDepth(3),
        chainDepthRange(-1),
        eeType ({app, enrol})
      }),
      PsidGroupPermissions ( WITH COMPONENTS {...,
            subjectPermissions (WITH COMPONENTS{
          explicit (SequenceOfPsidSspRange (SIZE (1)) (WITH COMPONENT (WITH COMPONENTS {
            psid (SecurityMgmtPsid), sspRange ABSENT
          })))
        }),
        minChainDepth(1),
        chainDepthRange(-1),
            eeType ({app, enrol})
      }),
      PsidGroupPermissions ( WITH COMPONENTS {...,
            subjectPermissions (WITH COMPONENTS{
          explicit (SequenceOfPsidSspRange (SIZE (1)) (WITH COMPONENT (WITH COMPONENTS {
            psid (MisbehaviorReportingPsid), sspRange ABSENT
          })))
        }),
        minChainDepth(1),
        chainDepthRange(-1),
            eeType ({app, enrol})
      }),
      PsidGroupPermissions ( WITH COMPONENTS {...,
            subjectPermissions (WITH COMPONENTS{
          explicit (SequenceOfPsidSspRange (SIZE (1)) (WITH COMPONENT (WITH COMPONENTS {
            psid (CrlPsid), sspRange ABSENT
          })))
        }),
        minChainDepth(1),
        chainDepthRange(-1),
            eeType ({app, enrol})
      })
    })),
```

This indicates that:

— For the PSIDs indicating Security Management, Misbehavior Reporting, and CRL signing, the chain to the end-entity certificate shall be of length at least 1 and may be of any length greater than or equal to 1;

— For all other PSIDs, the chain to the end-entity certificate shall be of length at least 3 and may be of any length greater than or equal to 3.

7.4.4.3 Receiving [in CRL security profile]

Change value of Maximum Certificate Chain Length from 8 to 7.

Note: the definition of chain length used by the security profile is changed by this guidance note, so "length 7" in this case means the same as "length 8" did in 1609.2-2016, i.e. that there are 8 certificates in total in the chain (and 7 inter-certificate gaps).

7.4.4.4 Security Management [in CRL security profile]

Change value of Maximum Certificate Chain Length from 8 to 7. Same reasoning as for 7.4.4.3.

9.2.2.5 Transition to *Key and Certificate* state [in 9.2.2, Cryptomaterial Handle]

Change

if the certificate is not part of a valid certificate chain terminating in a trust anchor

to

if the certificate is not part of a valid certificate chain beginning with a trust anchor

In all security profiles

Change " Maximum Certificate Chain Length" ***to*** " Maximum Full Certificate Chain Length"

D.3.1 Examples

Change all instances of " of length 2" ***to*** " of length 1, i.e. containing two certificates".